Алгоритмы на массивах

Алексей Владыкин

СПбГУ ИТМО

27 октября 2010

Представление массива в памяти

- Одномерный массив — непрерывный участок памяти
- Двумерный массив — либо один непрерывный участок, либо массив указателей на отдельные одномерные массивы
- N-мерные массивы — те же два варианта, но на каждую пару соседних измерений

Адресация элемента массива

- a[N]:

Адресация элемента массива

- a[N]:

$$addr(a[i]) = addr(a) + i \cdot sizeof(type)$$

Адресация элемента массива

- a[N]:

$$addr(a[i]) = addr(a) + i \cdot sizeof(type)$$

- a[M, N]:

Адресация элемента массива

- a[N]:

$$addr(a[i]) = addr(a) + i \cdot sizeof(type)$$

- a[M, N]:

$$addr(a[i, j]) = addr(a) + (i \cdot N \cdot +j) \cdot sizeof(type)$$

Адресация элемента массива

- a[N]:

$$addr(a[i]) = addr(a) + i \cdot sizeof(type)$$

- a[M, N]:

$$addr(a[i, j]) = addr(a) + (i \cdot N \cdot +j) \cdot sizeof(type)$$

- $a[N_1, \ldots, N_k]$:

Адресация элемента массива

- a[N]:

$$addr(a[i]) = addr(a) + i \cdot sizeof(type)$$

- a[M, N]:

$$addr(a[i, j]) = addr(a) + (i \cdot N \cdot +j) \cdot sizeof(type)$$

- $a[N_1, \ldots, N_k]$:

$$addr(a[i_1, \ldots, i_k]) = addr(a) + \sum_{j=1}^{k} \left(i_j \prod_{h=j+1}^{k} N_h \right) \cdot sizeof(type)$$

Основные алгоритмы

- Обмен двух элементов местами
- Последовательный просмотр
- Поиск в упорядоченном массиве
- Циклический сдвиг
- Сортировка

Обмен

```
t = a[i];
a[i] = a[j];
a[j] = t;
```

```
a[i] = a[i] ^ a[j];
a[j] = a[i] ^ a[j];
a[i] = a[i] ^ a[j];
```

```
a[i] = a[i] + a[j];
a[j] = a[i] - a[j];
a[i] = a[i] - a[j];
```

Последовательный просмотр

- Просмотр всех элементов по порядку, время $O(N)$
- Вывод массива на экран
- Вычисление суммы элементов, среднего значения
- Поиск минимума, максимума
- Поиск заданного значения
  (и вариант реализации с барьером)

Listing 1: C

```
for (i = 0; i < N; ++i) {
    // do something with a[i]
}
```

Listing 2: Java

```
for (Object obj : array) {
    // do something with obj
}
```

Поиск в упорядоченном массиве

- Некоторые задачи можно решить быстрее
- Поиск минимума, максимума, k-й порядковой статистики — прямой доступ за $O(1)$
- Поиск заданного значения — двоичный поиск за $O(\log_2 N)$

Поиск в упорядоченном массиве

- Некоторые задачи можно решить быстрее
- Поиск минимума, максимума, k-й порядковой статистики — прямой доступ за $O(1)$
- Поиск заданного значения — двоичный поиск за $O(\log_2 N)$

```
int l = 0, r = N;
while (l < r) {
    int m = (l + r) / 2;
    if (a[m] < x) {
        l = m + 1;
    } else {
        r = m;
    }
}
if (r < N && a[r] == x) {
    printf("found %d at a[%d]\n", x, r);
}
```

Циклический сдвиг

Задача (циклический сдвиг влево на k): переставить элементы массива a в порядке

$$a_k,\ a_{k+1},\ \ldots,\ a_{N-1},\ a_0, a_1,\ \ldots,\ a_{k-1}$$

Циклический сдвиг

Задача (циклический сдвиг влево на k): переставить элементы массива a в порядке

$$a_k,\ a_{k+1},\ \ldots,\ a_{N-1},\ a_0, a_1,\ \ldots,\ a_{k-1}$$

Алгоритм 1:

- k раз сдвинуть массив на один элемент влево — $O(Nk)$.

Циклический сдвиг

Задача (циклический сдвиг влево на k): переставить элементы массива a в порядке

$$a_k,\ a_{k+1},\ \ldots,\ a_{N-1},\ a_0, a_1,\ \ldots,\ a_{k-1}$$

Алгоритм 1:

- k раз сдвинуть массив на один элемент влево — $O(Nk)$.

Алгоритм 2:

- сдвинуть $\gcd(N, k)$ цепочек на k влево — $O(N)$.

Циклический сдвиг

Задача (циклический сдвиг влево на k): переставить элементы массива a в порядке

$$a_k,\ a_{k+1},\ \ldots,\ a_{N-1},\ a_0, a_1,\ \ldots,\ a_{k-1}$$

Алгоритм 1:

- k раз сдвинуть массив на один элемент влево — $O(Nk)$.

Алгоритм 2:

- сдвинуть $\gcd(N, k)$ цепочек на k влево — $O(N)$.

Алгоритм 3 (Керниган и Плоджер):

- перевернуть $a[0..N-1]$, затем $a[0..N-k-1]$ и $a[N-k..N-1]$ — $O(N)$.

Циклический сдвиг

Задача (циклический сдвиг влево на k): переставить элементы массива a в порядке

$$a_k,\ a_{k+1},\ \ldots,\ a_{N-1},\ a_0, a_1,\ \ldots,\ a_{k-1}$$

Алгоритм 1:

- k раз сдвинуть массив на один элемент влево — $O(Nk)$.

Алгоритм 2:

- сдвинуть $\gcd(N, k)$ цепочек на k влево — $O(N)$.

Алгоритм 3 (Керниган и Плоджер):

- перевернуть $a[0..N-1]$, затем $a[0..N-k-1]$ и $a[N-k..N-1]$ — $O(N)$.

Алгоритм 4 (кольцевой буфер):

- назначить $a[k]$ началом массива — $O(1)$.

Упражнения

Обойти двумерный массив $a[M, N]$:

- по строкам;
- по столбцам;
- горизонтальной змейкой;
- вертикальной змейкой;
- диагональной змейкой;
- по спирали.

Упражнение

Найти ошибку в алгоритме обращения массива:

```
int N;
int a[N];
for (i = 0; i < N; ++i) {
    int t = a[i];
    a[i] = a[N - 1 - i];
    a[N - 1 - i] = t;
}
```

Рекомендуемая литература


Бентли Дж.
Жемчужины программирования. 2-е изд.
СПб.: Питер, 2002. — 272 с.: ил.


Вирт Н.
Алгоритмы и структуры данных: Пер. с англ. — 2-е изд., испр.
СПб.: Невский Диалект, 2001. — 352 с.: ил. // Глава 1


Столяр С. Е., Владыкин А. А.
Информатика: Представление данных и алгоритмы.
СПб.: Невский Диалект; М.: БИНОМ. Лаборатория знаний, 2007. — 382 с.: ил. // Глава D


Шень А.
Программирование: теоремы и задачи.
М.: МЦНМО, 2004. — 296 с.: ил.